社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－OS－108 (22)
2008／4／24

省電力MIPSプロセッサにおけるOSの試作とシミュレーションによる電力評価

砂田徹也[†1] 関直臣[†2] 中田光貴[†3]
香嶋俊裕[†3] 近藤正章[†4] 天野英晴[†2]
宇佐美公良[†3] 中村宏[†4] 並木美太郎[†1]

本報告は，省電力化技術であるパワーゲーティングを細粒度に施したMIPS R3000ベースのCPUチップGeyser-0に対して，その上で動作するOSの試作およびそのOSを用いてシミュレーションによる電力評価を行う．細粒度は，ALU, SHIFT, MULT, DIVおよびCP0のそれぞれをPGの対象とすることを意味する．試作したOSは，例外・割り込み管理機能，システムコール，タスク管理機能を備えており，タイマ割込みによるマルチタスクを実現する．試作OSによって4種類のベンチマークプログラムをマルチタスクで動作させた際の電力評価を行った結果，総電力で平均約50%，リーク電力で平均約72%の消費電力削減を実現した．

Prototyping of Operating System for Power-saving MIPS Processor and its Evaluation by Simulations

Tetsuya Sunata,[†1] Seki Naomi,[†2] Mitsutaka Nakata,[†3]
Toshihiro Kashima,[†3] Masaaki Kondo,[†4] Hideharu Amano,[†2]
Kimiyoshi Usami,[†3] Hiroshi Nakamura[†4] and Mitaro Namiki [†1]

This paper describes prototype of OS for processor Geyser-0 based on MIPS R3000 with a fine grain power gating technique to reduce power consumption and evaluation of power by the simulations when running the OS. The processor has power gating units such as ALU, SHIFT, MULT, DIV, and CP0. The prototype OS has the exception management, the system call, and the task management, and its OS achieves the multitask by the timer interruption. As the results of evaluation running four benchmark program on the OS and the processor, 50% working power and 72% leakage power are reduced.

1. はじめに

近年，高性能なシステムLSIは高度情報化社会を支える基盤として広く利用されているが，その性能と消費電力はトレードオフの関係となるため，LSIの高性能化は，消費電力の面で限界にきている．そこでソフトウェア，ハードウェアの各分野で省電力技術の確立が大きな課題となっている．省電力技術の対象となる電力は大きく分けて，ダイナミック電力とリーク電力の2種類に分けられそれぞれの電力に対してもまた様々な研究がなされている．最近のプロセッサの傾向として，プロセスルールの微細化が進んでおり，90nm世代以降のチップでは，ダイナミック電力に対するリーク電力の増大が顕著に現れてきている．このため，プロセッサのリーク電力削減技術の確立は，省電力技術において大きなテーマの一つとなっている．

リーク電力の削減技術には，パワーゲーティング，Dual Vth，基板バイアスなどが挙げられる．この中でパワーゲーティングとは，プロセッサ内の回路の一部に対して電源供給を断ち，その回路をスリープさせることにより，リーク電力を削減する技術である．特に，そのスリープ期間の制御などをOSとアーキテクチャが連携することにより，省電力効果を見込むことができると考えられる．しかし，従来のパワーゲーティング[2)5)]では，その対象がマルチコアにおけるコア単位であったり，長い期間におけるスリープのみであるなど，時間的・空間的に荒い粒度で適用されてきた．

†1 東京農工大学
Tokyo University of Agriculture and Technology
†2 慶應義塾大学
Keio University
†3 芝浦工業大学
Shibaura Institute of Technology
†4 東京大学
The University of Tokyo

また, OS 単体による省電力技術として, ページマイグレーションを用いたメモリ管理の研究[3] やスケジューリング方法に着目したものなど, 多数の研究が行われている. 一方で, アーキテクチャと OS とが連携して, さらなる省電力を見込む研究も盛んである. 近年省電力技術として注目されている DVFS(ynamic Voltage and Frequency Scaling) は, ダイナミック電力の削減手法であり, DVFS と OS の協調[4] により, DVFS をより細かく制御することによる省電力の研究がおこなわれている.

そこで, 本研究プロジェクト[1] では細粒度パワーゲーティングを行うことが可能なプロセッサのプロトタイプ Geyser-0 を試作した. 細粒度とは, パワーゲーティングを CPU 内部の比較的小さな単位に適用すること, またスリープの期間も 1 命令ごとに適用可能であることを意味する. この Geyser-0 と OS とが協調することによって, 細粒度パワーゲーティングを実行状態を考慮した動的制御が可能となり, 更なる省電力効果を期待することができる.

本稿では, Geyser-0 の概要, そして Geyser-0 上で動作する OS の試作とその OS を用いてのシミュレーションによる電力評価結果について述べる.

なお, Geyser-0 は 3 月にチップが完成した直後であるため, 細粒度パワーゲーティングによる省電力効果の検証はシミュレーションによって行う.

2. Geyser-0 の概要

Geyser-0 の実装は, おもにプロジェクトの回路設計およびアーキテクチャグループによって行われた[1)7)8)9)]. 筆者はシステムソフトウェアグループとして, TLB や CP0 など OS と深く関係のある部分の設計の一部と, チップ全体の動作検証, OS の試作を行った. 本章では, Geyser-0 の全体像およびその特徴である細粒度パワーゲーティングについて述べる.

2.1 Geyser-0 の基本構成

Geyser-0 は, MIPS R3000 をベースとした標準的な 5 段パイプラインであり, その動作周波数を 200MHz として設計された. 図 1 に, Geyser-0 のコアと TLB/キャッシュの構成, およびパワーゲーティングの概要図を示す. 図中の Geyser-0 Core の点線で囲まれた部分が細粒度パワーゲーティングの対象となる. Geyser-0 では, TLB およびキャッシュはパワーゲーティングの対象とはなっていない.

Geyser-0 では, 従来のパワーゲーティングと比べて非常に粒度の細かい制御を実現している. パワーゲーティングの対象は, Geyser-0 がベースとしている

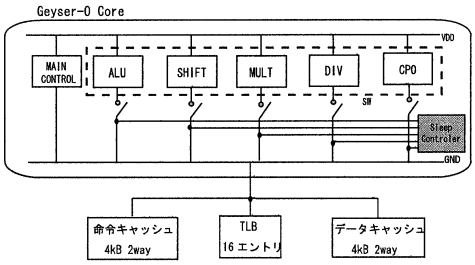

図 1 Geyser-0 の基本構成とパワーゲーティングの概要

MIPS R3000 が持つ演算器および, CPU と OS とのインタフェースの役割を持つ CP0 である. また, 1 命令ごとの細かな単位で制御が可能である.[7)] 図のように, Geyser-0 のパワーゲーティング回路の概要を示す. 各演算器に対して動作状態を切り替えるスイッチを挿入し, そのスイッチをスリープ制御信号を生成するスリープコントローラによって, パワーゲーティング制御を行う.

2.2 パワーゲーティングのスリープ対象

細粒度パワーゲーティングの基本動作は, 各演算器を使用する際にその演算器を動作状態へと切り替え, 演算が終了すると同時にスリープ状態へと移行する. Geyser-0 のパワーゲーティングによってスリープする対象を以下に示す.

- ALU Unit
  加減算などを行う一般的な演算器. 分岐命令, NOP 命令, またメモリアクセスでアドレス計算が不要な場合においてスリープする.
- SHIFT Unit
  シフト演算を行う演算器. シフト演算を行う時のみ, 動作状態となりそれ以外はスリープする.
- MULT Unit
  乗算を行う演算器. 乗算は 4 サイクル必要とする. 乗算においては, 二つのオペランドの上位 16 ビットが両方とも 0 であった場合には, 上位ビットの演算の必要がなくなるため, 上位部演算部を個別にスリープを行う.
- DIV Unit
  除算を行う演算器. 除算時のみ動作状態に切り替わり, それ以外はスリープする. 除算には 10 サイクル必要とする.
- CP0
  CP0 は, MIPS プロセッサ特有のコプロセッサであり, CP0 は CPU と OS とのインタフェース的な役割を持つ. CP0 はおもに以下の機能を持つ.
  – プロセッサのユーザモード, 特権モードの設定

– 例外発生時の原因情報の保持
– 割込みなどの例外処理
– 特権命令の処理
– メモリ管理機能
仮想-物理アドレス変換に必要な TLB エントリに関する情報の保持

特に, OS を導入した一般的なシステムにおいてユーザプログラムの実行時間は, OS が動作する時間よりも非常に長くなる. よって, カーネルが動作する特権モードの期間のみ動作状態へと切り替わる CP0 は, OS を導入することにより長期間のスリープが行え, それによる省電力効果を見込むことができる.

CP0 は演算器ではないが, 本稿では CP0 も含めて上記五つを演算器と呼ぶ.

2.3 細粒度パワーゲーティングと OS とのインタフェース

細粒度パワーゲーティングを制御するにあたって, OS に対して与えられているインタフェースとして, CP0 レジスタにパワーゲーティング制御用の PGStatus レジスタがある. パワーゲーティングによるスリープ, 動作状態の遷移には一定の電力が必要となり, またスリープ状態に切り替わっても一気にリーク電力が下がるわけではなく, 徐々に下がっていく傾向がある. ここで, 常に動作状態であった場合に消費する電力よりも状態遷移によるオーバヘッドを考慮してそれでもスリープ状態へ移行した方が, 消費電力が削減できる点 (サイクル数) をブレークイーブンポイントと呼ぶ.

PGStatus レジスタは, ブレークイーブンポイントに対して, OS 側が動的に対処するために設計された. 具体的にはパワーゲーティングの対象となる各演算器に対して通常とは異なるスリープポリシーを適用する際に用いる. スリープポリシーは以下の三つがある.

- 動的にスリープ制御 (通常)
- キャッシュミス時にのみスリープ
- 常に動作 (パワーゲーティングを行わない)

図 2 に, PGStatus レジスタを示す. PGStatus レジスタは, 各演算器のスリープポリシーを変更する際に, OS によって書き換えることが可能である. また, 動的

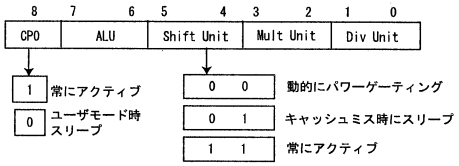

図 2 PGStatus レジスタ

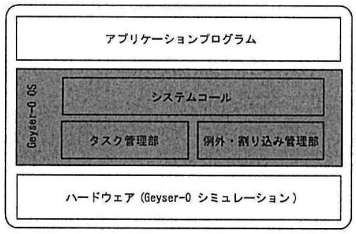

図 3 Geyser-0 OS の基本構成

スリープ制御を行わない場合でも, キャッシュミス時にのみスリープを行うかどうかを設定することもできる. これはキャッシュミスによって約 20 サイクルの遅延が発生するため, 一つのスリープポリシーとして用意した. スリープポリシーが 3 種類あるので, PGStatus レジスタの一つの演算器あたり 2 ビットが必要となる. ただし, CP0 に関しては, ユーザモード時にスリープを行うかどうかを決定すればよいので 1 ビットとなる. したがって PGStatus レジスタは 9 ビットで構成される.

3. Geyser-0 OS の試作

本研究は, 最終的には Geyser-0 上で Linux を実行し, より実用的な環境において評価を行うことを目標としている. そこで, まず単純な構造の OS を試作, 評価することによってパワーゲーティングおよびそれを OS で制御することの有効性を検証する. 簡易的なマルチタスク OS を試作し, それに対して電力評価を行うことで Geyser-0 で OS を動作させた際の基盤となる電力評価を求めることを目的とする.

3.1 Geyser-0 の設計方針

Geyser-0 OS は, 筆者の研究室で開発されてきた組み込み向け OS「開聞」をもとに試作した. 「開聞」[6] は, 軽量なタスク管理, 例外管理などを備えた OS となっている. 「開聞」は過去に MIPS プロセッサ (Vr4300) へと移植が行われており, Geyser-0 がベースとしている R3000 とは若干異なるものの MIPS アーキテクチャである点は変わらない. また「開聞」はリアルタイム OS としての動作を行うことも可能であり, 今後の発展的な部分として考えている. そこで筆者は, Geyser-0 の評価を行う際の試作 OS として「開聞」を参考とした.

3.2 Geyser-0 OS の基本構成

Geyser-0 OS の基本構成を図 3 に示す. Geyser-0 OS は, 簡易的なマルチタスク OS を実現するために

- 例外・割り込み管理部
- システムコール
- タスク管理部

のそれぞれの機能を有している．さらにCP0のスリープ効果を得るために，ユーザタスクはユーザモードで実行し，OSによって行う必要のある処理のみを特権モードで実行する．またVerilog HDLによってシミュレーションされるGeyser-0は，タイマが存在しないためタイムスライスによるマルチタスクの実現が不可能である．そこで，Verilogによってタイマを実装し，定期的にタイマ割り込みをGeyser-0にかける機構を試作した．

3.3 Geyser-0 OSの機能

3.3.1 例外・割り込み管理部

例外・割り込み管理部では，Geyser-0が発生させる例外に対してOS側でハンドリングする機能を提供する．またタイマなどの外部割り込みについてもGeyser-0では，例外として扱う．Geyser-0 OSは，現在のところ以下の例外・割り込みに対して例外処理を行う．

- タイマ割込み
  タイマは，今回のシミュレーションによる評価のために，試作したものを使用する．そのタイマによって定期的に割り込みが発生し，OSはタスク管理部を呼び出すことで，タスクスケジューリングによって決定された次に実行するべきタスクに対して，CPUの実行権を与える．
- TLBミス例外
  Geyser-0では，メモリ管理にTLBを使用しておりメモリアクセスを行う際には，TLBに対して参照が行われる．TLBは仮想アドレスと物理アドレスの対応表を保有している．あるメモリアクセスの際に，仮想アドレスに対する物理アドレスの対応がTLBエントリ(16エントリ)に存在しない場合，TLBミスが発生する．この場合OSは，アドレス変換に失敗した仮想アドレスに対して，適切な物理アドレスを取得し，その仮想-物理アドレスの対応関係を新たにTLBエントリに書き込むことで，TLBミスに対処する．
- システムコール例外
  Geyser-0では，ユーザタスクからのシステムコールにおいても例外を発生させる．OSは，呼び出されたシステムコールを判別し，適切な処理を行う．システムコールの詳細は，次節で示す．

3.3.2 システムコール

システムコールとして，表1を実装した． ユーザタスクは，これらタスク管理に必要なシステムコールを使用することができる．また，今回のGeyser-0 OSはシミュレーションで評価を行っているため，実I/O装置が存在しない．そこで，Verilog HDLの機能として用意されている文字列表示機能($display)を間接的にOSが使用することによって，文字列の出力を行う．入力に関しては，未対応でありこれはGeyser-0の評価を実機で行う際に実装を行う．

表1 システムコールの一覧

システムコール名	処理
sys_putstr	文字列の出力
sys_print	文字列，int型変数値の出力
Init_Task	初期タスクの生成
Create_Task	タスクの生成
Delete_Task	タスクの消去
Suspend_Task	タスクを実行可能状態に戻す
Resume_Task	タスクを実行可能状態に遷移
Yield	強制的にスケジューリングを呼ぶ

3.3.3 タスク管理部

タスク管理部は大きく分けて，以下の2種類の機能を提供している

- タスクの生成，消去機能
- スケジューリング機能

タスク管理部は，タスクをコンテキストやタスクIDなどいくつかの情報にまとめた，TCB(Task Control Block)によって管理する．コンテキストとして，Geyser-0に用意されている32個の汎用レジスタおよび乗算/除算の結果を格納する2個のレジスタ，そしてCP0の各レジスタの値を保持する．

それらを実行状態，実行可能状態，待ち状態の各キューに繋げる．タイマ割り込みが発生すると，OSは実行状態にあるタスクを実行可能状態のキューへと戻し，実行可能状態のキューから次に実行するべきタスクを取り出し実行状態とする．この動作によりマルチタスクを実現する．次に実行するべきタスクの決定を決めるスケジューリング方法は，各TCBには優先度を設けることで優先度スケジューリングを行う．

4. シミュレーションによる電力評価方法

4.1 Geyser-0の動作環境

Geyser-0は，東京大学大規模集積システム設計教育研究センター(VDEC)が提供するCADツールを使用して設計されており，Verilog HDLで記述されたGeyser-0をCadence社のNC-verilogと呼ばれる高速なverilogシミュレーションツールを用いて動作させることができる．今回試作したOSの動作や電力評価は，VerilogHDLで記述されたシミュレーションされるGeyser-0上での話として進める．

Geyser-0のシミュレーション環境では，以下の四つのファイルをシミュレーション時に用意する．

- ユーザプログラム

- OS プログラム
- ブートプログラム
- データファイル

それぞれのプログラム, データは, 仮想的に用意されたメモリ空間へと実行時に配置される. Geyser-0 のシミュレーションが始まると, Geyser-0 におけるリセットベクタから実行され, その部分にはブートプログラムがロードされる.

読み込ませる各プログラムは, まず LinuxPC 環境において, 開発したプログラムを MIPS クロスコンパイラとして準備した GCC(4.2.2) を用いて, MIPS 用にクロスコンパイルする. そしてコンパイル後のオブジェクトデータに対して, GCC のサブセットである objdump を用いて逆アセンブルを行い, 機械語を Verilog シミュレーション用に整形することで, 生成している. Verilog によるシミュレーションにはいくつかの段階が存在するが, 今回の OS シミュレーションは, RTL(Register Transfer Level) シミュレーションによって行う. また電力評価においてはさらに詳細なシミュレーションであるゲートレベルシミュレーションを行っている. RTL, ゲートレベル各シミュレーションを行う際に, ユーザプログラムや OS プログラムの生成方法に違いはない.

4.2 電力評価環境とベンチマークプログラム

電力評価は, OS プログラムによって生成された二つのベンチマークプログラムを, タイマ割込みによるタイムスライスで切り替えながら実行し, ベンチマークプログラムが終了するまでの電力を計測する. 二つのベンチマークプログラムは, それぞれ同じものを動作させる. またスリープポリシーは, 動的にパワーゲーティングを行う通常のポリシーとした.

今回ベンチマークプログラムは, 組込み系プロセッサ向けベンチマーク群である MiBench より, 数学演算パッケージから Quick Sort, ネットワークパッケージから Dijkstra を使用する. また行列演算である Matrix および整数演算ベンチマークとして Dhrystone をベンチマークプログラムとして採用した.

4.3 評価対象の電力

細粒度パワーゲーティングによって, 各演算器は動作状態とスリープ状態の遷移を繰り返して実行される. またそれぞれのモードにおいてダイナミック電力とリーク電力が発生する. ダイナミック電力とは, 回路動作に要する電力であり, リーク電力は, トンネル効果によるリーク電流によって発生する電力である. その際に, 図 4 のような電力推移が発生する. 図 4 の A〜D のそれぞれの記号を以下に説明する.

- 領域 A : 動作時電力
  動作時のダイナミック, リーク電力
  → 解析方法 : Power Compiler
- 領域 B : スリープ開始オーバーヘッド電力
  動作状態からスリープ状態への移行に必要な電力.
  → 解析方法 : Power Compiler
- 領域 C : スリープ時電力
  スリープ時のリーク電力. スリープ時のダイナミック電力は 0 とみなす.
  → 解析方法 : プロジェクト内で開発された電力評価ツール[9)]
- 領域 D : 動作状態への切り替え時オーバーヘッド電力
  領域 B とは逆にスリープ状態から動作状態に移行する際に必要な電力
  → 解析方法 : プロジェクト内で開発された電力評価ツール

A〜D の各領域において, 動作時の平均電力, スリープ時の平均電力を各ツールを用いることで求め, 電力評価を行う.

4.4 動作時ダイナミック電力およびリーク電力の取得方法

領域 A の電力である, 動作時のダイナミック電力およびリーク電力の取得方法には, Synopsys 社の Power Compiler というツールを用いる. Power Compiler による電力評価を行うには, SAIF と呼ばれるトランジスタのスイッチング確率を取得する必要があり, その取得にゲートレベルシミュレーションを行う.

また領域 B のスリープ開始オーバヘッド電力だが, これについても領域 A と同様に SAIF を用いて Power Compiler の解析によって求められる.

4.5 スリープ時リーク電力の取得方法

スリープ時の電力 (領域 C) だが, ここでスリープ時のダイナミック電力は, 動作していない状態なので 0 と近似する. したがってここではスリープ時のリーク

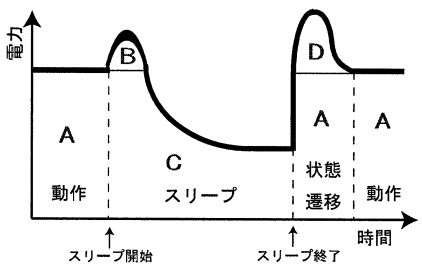

図 4 評価対象となる電力

電力の取得方法について述べる．

まず前準備として細粒度パワーゲーティングの対象のスリープ/動作状態それぞれ頻度情報を取得する．ここで必要となる頻度情報とは，

N サイクルのスリープが M_N 回あった

という情報である．これは RTL シミュレーションによって取得できる．この頻度情報をもとに，領域 C, D の平均消費電力を求める．

領域 C は，これはスリープ状態遷移の際の過渡状態の電力である．この電力を Synopsys 社の HSIM というツールを用いて取得し，N サイクルを連続スリープさせた場合の電力評価モデルを構築する．この評価モデルを用いて，ある演算器が N サイクルスリープした後に動作状態へ移行する場合の消費オーバヘッド電力 L_{ws_N}(領域 D) を算出する．そして，領域 C と D の値から，スリープ時の平均リーク電力を求める．

N サイクルのスリープが M_N 回あり，アプリケーションの総サイクル数が T サイクルだとすると，次の式で，各演算器のスリープ時の平均リーク電力 L_{ws} を算出することできる．

$$L_{ws} = \sum_i (L_{ws_i} * \frac{M_i * i}{T}) \quad (1)$$

また Power Compiler によって求められた動作時の平均リーク電力を L_{wa} とすると，(1) で求めた L_{ws} とを合わせて，シミュレーション全体における平均リーク電力 L_w を以下の式で求めることができる．

$$L_w = L_{ws} + L_{wa} * (1 - \frac{\sum_i (M_i * i)}{T}) \quad (2)$$

本研究では，今回プロジェクト内で開発された電力解析ツールを用いている．[9] これは HSIM による解析結果をデータベース化することで，使用頻度情報の解析のみで領域 C, D の消費電力の見積もりを行うことができる．

4.6 電力評価モデル

ここまで挙げた各電力をもとに電力評価のモデルを構築する．ここで評価に用いる電力は，上記に示す方法で求めることが可能な四つの電力である．

- $\bar{D}^i_{Act}$:動作時の平均ダイナミック電力
- $\bar{L}^i_{Act}$:動作時の平均リーク電力
- $\bar{D}^i_{Slp}$:スリープ時の平均ダイナミック電力 ($= 0$)
- $\bar{L}^i_{Slp}$:スリープ時の平均リーク電力

ここで $i \in alu, shift, mult, div, cp0$ である．

演算器 i の平均消費電力を $\bar{W}^i$ とすると

$$\bar{W}^i = (\bar{D}^i_{Act} + \bar{L}^i_{Act}) * \frac{C^i_{ON}}{C_{ALL}} + (\bar{D}^i_{Slp} + \bar{L}^i_{Slp}) * \frac{C^i_{OFF}}{C_{ALL}}$$

となる．C^i_{ON} は，演算器 i が ON となるサイクル数．C^i_{OFF} は，演算器 i が OFF となるサイクル数である．また C_{ALL} は，シミュレーションの総サイクル数である．そして，演算器全体の総平均消費電力 $\bar{W}^{ALL}$ は，演算器ごとの電力を合計して

$$\bar{W}^{ALL} = \sum_i \bar{W}^i [W] \quad (3)$$

となる．またプログラムの実行時間を T_{ALL} とすると，その演算器全体が消費するエネルギー E_{ALL} は，

$$\begin{aligned} E_{ALL} &= \int_0^{T_{ALL}} \bar{W}_{ALL} dt \\ &= \bar{W}_{ALL} * T_{ALL} [J] \end{aligned} \quad (4)$$

となる．ここで，T_{ALL} は，Geyser-0 の動作周波数が 200MHz を想定しているので，サイクル数で表現すると，1[サイクル] = 5[ns] となる．よって式 (4) は，C_{ALL} を用いて，以下のように書き換えることができる．

$$\begin{aligned} E_{ALL} &= \bar{W}_{ALL} * T_{ALL} \\ &= \bar{W}_{ALL} * 5 * 10^{-9} * C_{ALL} [J] \end{aligned} \quad (5)$$

5. 評価結果と考察

過去に OS を含まないデータは，プロジェクトの他メンバーの論文[7][8][9] にあった．そこで本稿では，OS を含んだ状態でのパワーゲーティングの効果を明らかにする．

5.1 各演算器の使用頻度

スリープ時のリーク電力算出に必要な各演算器の使用頻度を図 5 に示す．　各ベンチマークで，ALU と SHIFT の使用頻度が 80%を越えており使われる機会の多い演算器であることがわかる．一方 MULT や DIV は，今回用いたベンチマークではほとんど用いられておらず，行列演算を行っている MATRIX で MULT が約 26%の使用率であった．特権モードとして OS の処理を行う CP0 の利用率は，5%程度であった．つまり 95% は CP0 が使われないということとなる．この利用率から CP0 に対して，OS の導入による細粒度パワーゲーティングの効果が大きく現れると考

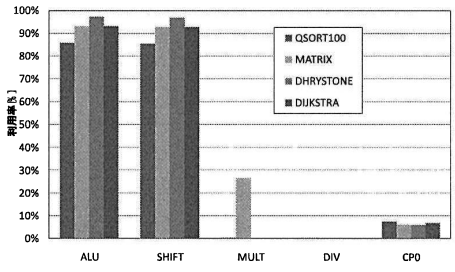

図 5　各演算器の使用頻度

えられる.

5.2 消費電力削減効果

本節では, パワーゲーティングの対象である演算器全体が消費する総電力/エネルギー, パワーゲーティングの効果が見込めるリーク電力の削減効果, および温度によるリーク電力の違いについて示す.

5.2.1 演算器の総消費電力/エネルギー

まず, 式 (4) で示される演算器全体が消費する総電力 (ダイナミック電力+リーク電力) を図 6 に示す. ここで, 図中の SUM(PG) は各演算器の消費電力を合計した値であり, SUM(NO_PG) は, パワーゲーティングを行わなかった場合での各演算器の消費電力である.

各ベンチマークについて, 平均約 50%に近い消費電力の削減をパワーゲーティングにより実現した. 次に消費エネルギーを求める. エネルギー算出 (式 (5)) に必要となる実行時間はシミュレーションの際のサイクル数より求めるため予想実時間と呼ぶ. サイクル数および予想実時間を表 2 に示す.

表 2 各アプリケーションごとの予想実時間 [単位: msec]

プログラム	総サイクル数 [Clock]	予想実時間 [msec]
QSORT100	395986	1.98
MATRIX	367646	1.84
DHRYSTONE	388650	1.94
DIJKSTRA	419164	2.1

これより算出した消費エネルギーを図 7 に示す. 図 7 は, 図 6 と同じ形を示しており, 総消費電力と同じく各ベンチマークにおいて, 平均約 50%の消費エネルギー削減を実現している. これは, 表 2 に示すサイクル数および, そこから求められる予想実時間がパワーゲーティングを行う前後で変わらないためである. つまり, パワーゲーティングによる性能低下は存在しないことがわかる.

5.2.2 演算器ごとのリーク電力

次に, パワーゲーティングによる効果が期待できる

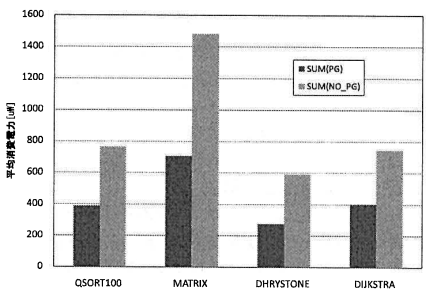

図 6 25 ℃での演算器の総消費電力

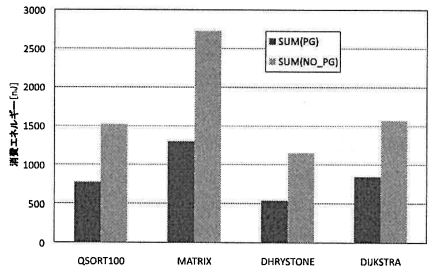

図 7 25 ℃での演算器の総消費エネルギー

リーク電力の削減効果を示す. 図 8 に, 各ベンチマークごとの 25 ℃での平均リーク電力を示す. 図内の Active という値は, パワーゲーティングを行わなかった場合の各ベンチマークごとのリーク電力の平均である.

演算器ごとに比較を行うと, 各ベンチマークにおいて使用頻度の高い ALU や SHIFT は, リーク電力の削減の効果が少ない. しかし, 比較的使用頻度が低い MULT では MATRIX において約 40%, それ以外でも約 75%のリーク電力が削減, 全く使われない DIV では, 約 85%以上のリーク電力削減がなされている.

特に OS を導入する効果として着目する CP0 のリーク電力については, 各ベンチマークで約 80%近くのリーク電力が削減できた. これは OS によって特権モードとユーザモードを管理し, CP0 の使用率を 5%程度まで抑え約 95%の期間をスリープさせることができたためであり, OS による省電力効果が現れている.

各ベンチマークで平均を取ると, パワーゲーティングを施した状態でリーク電力が 42.7[uW] となる. また, パワーゲーティングを施していない状態でリーク電力は, 150.23[uW] となる. したがって演算器全体の平均で考えると $\frac{42.7}{150.23} * 100 \cong 28[\%]$ となり, パワーゲーティング対象において約 72%のリーク電力削減効

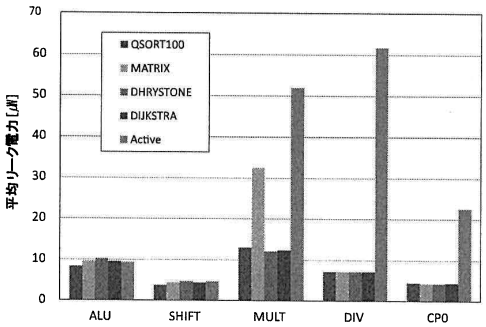

図 8 25 ℃での各演算器のリーク電力

果を得た.

5.2.3 温度の違いによるリーク電力

シミュレーションによる電力評価は, 温度の違いによる評価を行うことが可能である. 25 ℃と 100 ℃の二つの温度における, ベンチマークごとの演算器ごとのリーク電力の合計と, パワーゲーティングを行わなかった場合のリーク電力を図 9 に示す. 図内の Active という値は, パワーゲーティングを行わなかった場合の各ベンチマークごとのリーク電力の平均である. 100 ℃のリーク電力は, 25 ℃のリーク電力と比べて各ベンチマークで平均 2.1 倍増加した. またパワーゲーティングを行わない場合, 25 ℃に対して 100 ℃のリーク電力は, 約 10 倍の増加となる. 全ての演算器について温度が上がるとスリープ時のリーク電力以上に, 動作時のリーク電力が非常に大きくなる. 動作時のリーク電力が増えるということは, パワーゲーティングによるオーバヘッドを考慮したすると, ブレークイーブンポイントが温度が上がるにつれて短くなることを意味する.[7] これより, 消費電力やブレークイーブンポイントは非常に温度変化に敏感であることがわかる.

今回の電力評価は, ブレークイーブンポイントを意識せずに, スリープポリシーを固定して行った. したがって, OS によって PGStatus を用いて温度変化を考慮したパワーゲーティングの動的制御を行えば, より省電力効果を見込むことができる.

6. おわりに

本稿では, 細粒度パワーゲーティングを施した CPU コア Geyser-0 に対して, OS の試作およびその OS を用いての電力評価を行った. 結果, 演算器ごとの平均リーク電力で約 72%, OS による効果を期待した CP0 では約 80%のリーク電力が削減できた. また CPU コア全体では約 37%のリーク電力削減効果を得た.

今後の課題は, スリープポリシーの変更が可能な PGStatus レジスタを用いて, ブレークイーブンポイントにもとづいた細粒度パワーゲーティングの動的制御を行いさらなる省電力効果を得ること. そしてチップが完成したのでシミュレーションではなく, Geyser-0 実機での電力評価を行いたいと考えている.

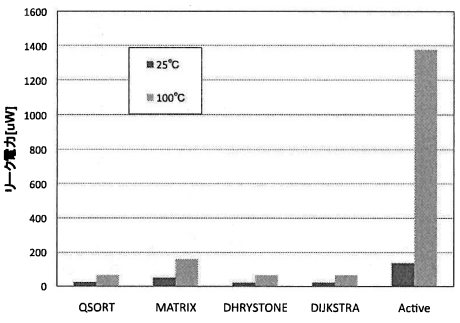

図 9 25 ℃と 100 ℃でのリーク電力

謝辞 本研究は東京大学大規模集積システム設計教育研究センター (VDEC) を通し, 株式会社半導体理工学研究センター, 富士通株式会社, 松下電器産業株式会社, NEC エレクトロニクス株式会社, 株式会社ルネサステクノロジ, 株式会社東芝の協力で行われたものである.

本研究は, 科学技術振興機構「JST」の戦略的創造研究推進事業「CREST」における研究領域「情報システムの超低消費電力化を目指した技術革新と統合化技術」の研究課題「革新的電源制御による次世代超低電力高性能システム LSI の研究」によるものである.

参考文献

1) 中村宏 他: 革新的電源制御による超低消費電力高性能システム LSI の構想, 情報処理学会研究報告 ARC-173, pp.79-84 (2007).
2) Zhigang Hu et al.: Microarchitectural Techniques for Power Gating of Execution Units, Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED 04). pp32-37 (2004).
3) Pratap Ramamurthy et al.: Performance-directed Energy Management using BOS, ACM SIGOPS Operating Systems Review, Vol.41, pp.66-77 (2007).
4) 金井 遵 他: 統計情報に基づく省電力 Linux スケジューラ, 情報処理学会「システムソフトウェアとオペレーティング・システム」研究会第 106 回, 2007 年並列/分散/協調処理に関する『旭川』サマー・ワークショップ (SWoPP 旭川 2007), Vol.2007-OS-106(2), pp.9-16 (2007).
5) Y.Kanno: Hierarchical Power Distribution with 20 Power Domains in 90-nm Low-Power Multi CPU Processor, ISSCC2006, pp.540-541 (2006).
6) 堀口 努, 萱嶋 志門, 並木 美太郎: 組込み用 OS『開闢』の MIPS プロセッサへの移植, 情報処理学会研究報告 OS-87, pp.57-64 (2001).
7) 関 直臣 他: MIPS R3000 プロセッサにおける細粒度動的スリープ制御の実装と評価, 情報処理学会技術研究報告 2007-ARC-176, pp.71-76, (2008)
8) 白井 利明 他: ランタイムパワーゲーティングを適用した MIPS R3000 プロセッサの実装設計と評価, 電子情報通信学会技術研究報告 VLD2007-112, pp.43-48, (2008)
9) 中田 光貴 他: ランタイムパワーゲーティングを適用した回路での検証環境と電力見積もり手法の構築, 電子情報通信学会技術研究報告 VLD2007-111, pp.37-42 (2008).